SESUG Paper 273-2018

Automation Methods: Using SAS® to Write the Code for Repetitive Quality Control Checks

Imelda C. Go, Questar Assessment, Inc.

ABSTRACT

This paper illustrates how you can use a SAS data set, which we will call a quality control specifications (QCS) data set, to store the information on how each variable in the data set will be scrutinized via QC methods. For example, we will need to run PROC FREQ for variables A to Z and run a PROC MEANS for variables X1-X100. Let us suppose we have an existing data set we need to run quality control (QC) checks on. We first apply PROC CONTENTS to this data set to get the complete list of variables in the data set. We then focus on adding to this data set the variables that identify which QC methods apply to each variable. Once this information has been added to the data set, this becomes our QCS data set. We then use SAS to process the information in it so that SAS creates the programming statements required to execute the QC checks. This simplifies the task tremendously for the programmer because the programmer can focus on making sure that each variable has been marked with all the required QC checks and be assured that SAS will create the code for you.

At one point or another, a programmer faced with mundane, boring, repetitive, and/or time-consuming manual work has to start thinking about how to automate one's work in order to save time, to prevent errors, and to maintain one's sanity. When you have repetitive code, macros are a natural solution.

From year-to-year, there might be slight changes in your QC checks. Many of us have been burned by "modifying the program from last year." Errors can slip in because of having to sift through so much code to make a few changes. Perhaps you didn't insert the new code in the logical place in the program so you might have misplaced the code or worse accidentally left it out. You might have cut-and-paste the code from another part of the program and updated the variables but forgot to update the titles. Manual updates on code has challenges.

Here is the paradigm shift. Instead of exerting so much effort in manually updating, focus on creating a data set that reflects the things that you want to put in the code and then let SAS create the code for you instead.

Here are the steps for a contrived example.

1. Let us suppose that you have two records in a data set called `testdata`. It contains the date of birth, last name, first name, score, and counter variable. The `ctr` variables is simply the _n_ value (positional record number in the data set) and corresponds to the OBS value in PROC PRINT.

dob	lname	fname	score	ctr
2007/01/15	Smith	Ann	100	1
2008/02/16	Doe	Jane	98	2

2. Run a PROC CONTENTS on the data set that you need to perform QC checks on. Without the KEEP statement, PROC CONTENTS will provide a multitude of variables for you. For this example, it suffices to keep the `varnum` and `name` variables for this example. The `varnum` variable is the variable number and the `name` variable is the name of the variable. If there are 10 variables in the data set, there will be 10 records in the `qcscontents` output data set. The `varnum` value will range from 1 to 10. `Varnum` is 1 for the 1st variable in the data set, it is 2 for the 2nd variable, etc. (Note: If you do not have a pre-existing data set, then you will have to type but be careful not to miss any variables. Using PROC CONTENTS output offers completeness of the list of variables.)

```
proc contents data=testdata (drop=ctr) out=qcscontents (keep=varnum name)
noprint;
```

Name	varnum
dob	1
fname	3
lname	2
score	4

3. For each variable in the data set, determine which QC checks will be conducted and create your QCS data set as shown in the example below.
    - `Name` has a value for each variable in `testdata` that you want to conduct QC checks on.
    - If you have a variable that does not have a QC check, then you can either remove it from the QCS or keep it there with all QC flags set to missing or unflagged. In this example, `QC1-QC3` are QC flags (Y/blank values).
    - QC check number 1 generates the full frequency table.
    - QC check number 2 generates the frequency table only for every 2nd record (i.e., records with an even record number in the data set). When there are a large number of records, there is sometimes just the need to look at sample values in the PROC FREQ output. This can be done via sampling through the modulo operator. The condition `if mod(ctr,2)=0 identifies` every 2nd record in the data set. It returns the remainder of dividing `ctr` with 2. The modulo divisor can be set to a value that is convenient for you.
    - QC check number 3 generates the PROC MEANS output.
    - In the example below, there is only one title (`qcvartitle`) for each variable. In reality, you might have a need to have a title per type of QC check. Therefore, once you have recorded in QCS everything you need for YOUR code, you are ready to write the macro that drives your QC checks.

varnum Variable number	**name Variable name**	**QC1 Full frequency table**	**QC2 Frequency table for every 2nd record**	**QC3 PROC MEANS**	**qcvartitle Variable's title in QC check**
1	lname		Y		Last name of every 2nd student
2	fname		Y		First name of every 2nd student
3	dob	Y			Check the DOB distribution
4	score	Y		Y	Check the range of the scores

Using the QCS data set has benefits:

- The QCS and accompanying macro document and summarize your QC work in SAS and can be easily updated/modified. The data succinctly answer the question: "What did you check for each variable?" Trying to answer this question by sifting through manually generated code will take you a while. When you think about it, you will need to have the QCS data in order to finish programming. So many of us have been transferring our mental knowledge of what needs to be done into the SAS code directly. The paradigm shift is to document this knowledge in the QCS data set and then use SAS to write the code that will execute the QCS data.
- Using this technique can reduce debugging time. By focusing on the QCS data set, you can focus on the quality of your input so that SAS creates the program for you without the fatigue and human error factor. When so many variables are involved and work is so dependent on manual work (your memory and your typing), you can easily make a mistake somewhere. As long as the QCS data set drives the checking, you can be sure the code necessary for the check was executed according to the macro you wrote. All you need to do is validate the macro and it will execute correctly for each record in QCS. If you do not use a macro, the sheer number of variables and volume of code can be psychologically oppressive.
- Let SAS dynamically create the code for you. Using this technique can generate a whole lot more programming code than some programmers are used to, but if the code is dynamically created for you, it should not be a problem. You won't accidentally not check something unless you intended not to. Your QCS and macro will run the checks in your QCS and macro specifications.

PROGRAMMING EXAMPLE

Using the testdata and QCS data sets above, we continue with the coding.

SAS CODE	EXPLANATION

`proc means data=qcs noprint;`
`var varnum; output out=numvars n=numvars;`
`run;`
 	
`data numvars;`
`set numvars;`
`call symput('numvars',numvars);`
`run;`
	First, we need to define a macro variable `numvars` that contains the number of variables in the QCS data set. This is one way of achieving this. `Varnum` is numeric and when used in PROC MEANS, its n variable will indicate how many `varnum` values are in the data set.

SAS CODE	EXPLANATION

`%macro qcchecks;`
`%do i=1 %to &numvars;`
`data temp;`
`set qcs (where=(varnum=&i));`
`call symput ('qcvarname',strip(name));`
`call symput ('qc1',qc1);`
`call symput ('qc2',qc2);`
`call symput ('qc3',qc3);`
`call symput ('qcvartitle',qcvartitle);`
`run;`
	We now start to write the macro.
	
	We first iterate macro variable 1 from 1 till the number of variables (`numvars`) in QCS. We use the WHERE option to isolate the information for each variable.
	
	A temporary data set, called `temp`, is created. (You can opt to use a `data _null_` step instead.) All the values in the data set are used to create macro variables important to the processing for a particular value. These macro variable values are then used to perform the QC check.

`title "&i &qcvarname -- &qcvartitle";` `run;`	A title is then created that contains the `varnum` value, the variable name (`qcvarname`), and the QC title (`qcvartitle`).

`%if &qc1=Y %then %do;`
`proc freq data=testdata;`
`tables &qcvarname;`
`%end;`
 	
`%if &qc2=Y %then %do;`
`proc freq data=testdata;`
`tables &qcvarname;`
`where mod(ctr,2)=0;`
`%end;`
 	
`%if &qc3=Y %then %do;`
`proc means data=testdata;`
`var &qcvarname;`
`%end;`
`run;`
`%end;`
 	
`%mend qcchecks;`
	The next lines produce the code for each type of QC check.
	
	For QC1=Y, the complete frequency table for the variable is produced.
	
	For QC2=Y, the frequency table for the variable is produced but only for every 2nd record.
	
	For QC3=Y, the PROC MEANS output is produced for the variable.

`%qcchecks;` `run;`	The macro is then invoked and the QC checks for each variable are produced with a lot less effort than trying to generate the code manually.

There are a number of possibilities with this code. You can change the order of the data processing by how you write the macro. In this example, all the processing for a particular variable appear consecutively.

Programmers also group the output of variables according to the PROC first. For example, all the PROC FREQ output are provided for all the variables and then the output from another PROC invocation follows for variables of interest. It is just a matter of preference. If you want to iterate through the variables in QCS to run each type of QC check first, then you can rewrite your macro to do that.

The following were generated by the macro qcchecks.

1 lname -- Last name of every 2nd student

The FREQ Procedure

Lname	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Doe	1	100.00	1	100.00

2 fname -- First name of every 2nd student

The FREQ Procedure

Fname	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Jane	1	100.00	1	100.00

3 dob -- Check the DOB distribution

The FREQ Procedure

Dob	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2007/01/15	1	50.00	1	50.00
2008/02/16	1	50.00	2	100.00

4 score -- Check the range of the scores

The FREQ Procedure

Score	Frequency	Percent	Cumulative Frequency	Cumulative Percent
98	1	50.00	1	50.00
100	1	50.00	2	100.00

4 score -- Check the range of the scores

The MEANS Procedure

Analysis Variable : score				
N	Mean	Std Dev	Minimum	Maximum
2	99.0000000	1.4142136	98.0000000	100.0000000

CONCLUSION

This automation method proposes a way of programming that has the potential to decrease human error especially when many variables and QC checks are involved. The programmer will document the QC checks in the QCS data set and then use SAS to write the macro code that will operate according to the contents of the QCS data. The method also offers the advantage that the QCS data set documents all the QC checks that the programmer was supposed to or intended to check.

CONTACT INFORMATION

Imelda C. Go
igo@questarai.com
Working remotely from Columbia, SC

TRADEMARK NOTICE